CHAPTER 1

INTRODUCTION

1.1 History of Glaucoma

Glaucoma has been recognized for thousands of years, with its understanding evolving from ancient theories to modern scientific advancements. The term *glaucoma* comes from the Greek word *glaukos*, meaning “bluish-green,” likely referring to the cloudy appearance of the eye in advanced stages.

Ancient Egyptian and Greek physicians, including Hippocrates (460–370 BCE)**,** described vision loss conditions but did not differentiate glaucoma from cataracts. Roman physician Galen (129–216 CE) mentioned optic nerve damage, and medieval scholars like Avicenna (980–1037 CE) noted symptoms resembling glaucoma. However, the connection between increased intraocular pressure (IOP) and the disease remained unknown.

The 17th to 19th centuries saw clearer distinctions between glaucoma and cataracts. In 1850, Albrecht von Graefe identified IOP as a key factor and introduced iridectomy as a surgical treatment. Later, Frans Cornelis Donders advanced the understanding of glaucoma’s physiological effects.

The 20th century brought major breakthroughs, including the Scholtz tonometer (1905) for measuring IOP and modern surgical techniques like trabeculectomy and laser trabeculoplasty**.** Today, AI-based methods, optical coherence tomography (OCT), and deep learning models significantly enhance early detection, helping to prevent vision loss and blindness.

1.2 Need for a Glaucoma detector

Glaucoma is a leading cause of irreversible blindness worldwide, affecting millions of people, often without noticeable symptoms in its early stages. Since vision loss from glaucoma is permanent, early detection is crucial to prevent severe damage to the optic nerve. Traditional diagnosis relies on comprehensive eye exams performed by ophthalmologists, including intraocular pressure (IOP) measurement, visual field tests, and optic nerve imaging. However, these methods require specialized equipment, are time-consuming, and may not be accessible in rural or underdeveloped regions.

This project aims to address these challenges by developing an automated glaucoma detection system using digital image classification and Convolutional Neural Networks (CNNs)**.** By analyzing cropped retinal fundus images, the system can quickly and accurately classify eyes as glaucomatous or non-glaucomatous. The use of TensorFlow for deep learning ensures high model performance, while Streamlit enables a user-friendly web application for real-time diagnosis.

Such an AI-powered tool can assist ophthalmologists by providing an initial screening, reducing workload, and ensuring early intervention. It also makes glaucoma detection more accessible in remote areas where specialized medical professionals are scarce. Ultimately, this project contributes to improving healthcare by offering a fast, cost-effective, and efficient solution for glaucoma

CHAPTER 2
LITERATURE REVIEW

1. GLAUCOMA DETECTION USING DIGITAL IMAGE PROCESSING AND SUPERVISED LEARNING

Bhagyaveni M.A

A difficult challenge in the realm of biomedical engineering is the detection of physiological changes occurring inside the human body, which is a difficult undertaking. At the moment, these irregularities are graded manually, which is very difficult, time-consuming, and tiresome due to the many complexities associated with the methods involved in their identification. In order to identify illnesses at an early stage, the use of computer-assisted diagnostics has acquired increased attention as a result of the requirement of a disease detection system. The major goal of this proposed work is to build a computer-aided design (CAD) system to help in the early identification of glaucoma as well as the screening and treatment of the disease. The fundus camera is the most affordable image analysis modality available, and it meets the financial needs of the general public. The extraction of structural characteristics from the segmented optic disc and the segmented optic cup may be used to characterize glaucoma and determine its severity. For this study, the primary goal is to estimate the potential of the image analysis model for the early identification and diagnosis of glaucoma, as well as for the evaluation of ocular disorders. The suggested CAD system would aid the ophthalmologist in the diagnosis of ocular illnesses by providing a second opinion as a judgment made by human specialists in a controlled environment. An ensemble-based deep learning model for the identification and diagnosis of glaucoma is in its early stages now. This method’s initial module is an ensemble-based deep learning model for glaucoma diagnosis, which is the first of its kind ever developed. It was decided to use three pretrained convolutional neural networks for the categorization of glaucoma. These networks included the residual network (ResNet), the visual geometry group network (VGGNet), and the GoogLeNet. It was necessary to use five different data sets in order to determine how well the proposed algorithm performed. These data sets included the DRISHTI-GS, the Optic Nerve Segmentation Database (DRIONS-DB), and the High-Resolution Fundus (HRF). Accuracy of 91.11% for the PSGIMSR data set and the sensitivity of 85.55% and specificity of 95.20% for the suggested ensemble architecture on the PSGIMSR data set were achieved.

2. DEEP CONVOLUTIONAL NEURAL NETWORKSFOR ACCURATE DIAGNOSIS OF GLAUCOMA USING RETINAL FUNDUS IMAGES

U Raghavendra , Hamido Fujita , Sulatha V Bhandaary

Glaucoma progressively affects the optic nerve and may cause partial or complete vision loss. Raised intravascular pressure is the only factor which can be modified to prevent blindness from this condition. Accurate early detection and continuous screening may prevent the vision loss. Computer aided diagnosis (CAD) is a non-invasive technique which can detect the glaucoma in its early stage using digital fundus images. Developing such a system require diverse huge database in order to reach optimum performance. This paper proposes a novel CAD tool for the accurate detection of glaucoma using deep learning technique. An *eighteen* layer convolutional neural networks (CNN) is effectively trained in order to extract robust features from the digital fundus images. Finally these features are classified into normal and glaucoma classes during testing. We have achieved the highest accuracy of 98.13% using 1426 (589: normal and 837: glaucoma) fundus images. Our experimental results demonstrates the robustness of the system, which can be used as a supplementary tool for the clinicians to validate their decisions.

3. GLAUCOMA DETECTION USING IMAGE PROCESSING TECHNIQUES

Abdullah Sarhan , John Rokne , Reda Alhaj

The term glaucoma refers to a group of heterogeneous diseases that cause the degeneration of retinal ganglion cells (RGCs). The degeneration of RGCs leads to two main issues: (i) structural changes to the optic nerve head as well as the nerve fiber layer, and (ii) simultaneous functional failure of the visual field. These two effects of glaucoma may lead to peripheral vision loss and, if the condition is left to progress it may eventually lead to blindness. No cure for glaucoma exists apart from early detection and treatment by optometrists and ophthalmologists. The degeneration of RGCs is normally detected from retinal images which are assessed by an expert. These retinal images also provide other vital information about the health of an eye. Thus, it is essential to develop automated techniques for extracting this information. The rapid development of digital images and computer vision techniques have increased the potential for analysis of eye health from images. This paper surveys current approaches to detect glaucoma from 2D and 3D images; both the limitations and possible future directions are highlighted. This study also describes the datasets used for retinal analysis along with existing evaluation algorithms. The main topics covered by this study may be enumerated as follows:

- Approaches to segment different objects from both 2D and 3D images;
- Approaches that may lead to encouraging results for glaucoma detection;
- Challenges faced by researchers
- Currently available retinal datasets and evaluation method

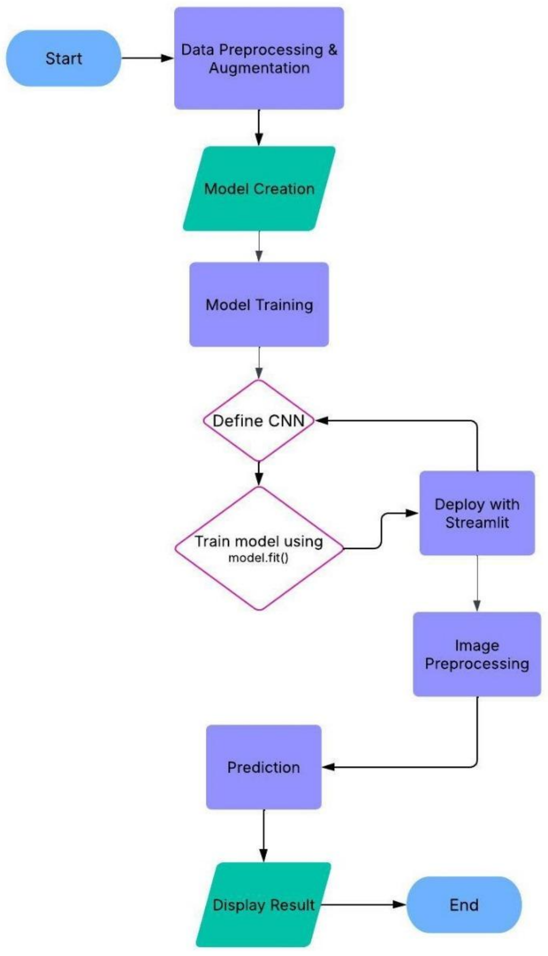
CHAPTER 3

FLOW CHART

1.1Introduction

The flow chart illustrates an automated glaucoma detection system using deep learning techniques. The process begins with acquiring fundus or OCT images, which are then preprocessed to enhance their quality through resizing, normalization, and augmentation. A Convolutional Neural Network (CNN) built with TensorFlow extracts crucial features from the images and classifies them as glaucoma or non-glaucoma. The classification output provides predictions along with confidence scores. Finally, the system is deployed using Streamlit, enabling users to upload retinal images and receive real-time predictions. This framework offers an efficient, accessible, and scalable approach to assist in early glaucoma detection and diagnosis.

1.2Flow chart



1.3 Explanation

The flowchart outlines the process of glaucoma detection using CNN and digital image classification, starting from data preprocessing to result display.

The process begins with the *"Start"* node, indicating the initiation of the pipeline. The first step, *"Data Preprocessing & Augmentation"*, involves preparing the dataset by cleaning, normalizing, and augmenting images to enhance model performance. This ensures the CNN receives diverse and high-quality input.

Next, *"Model Creation"* refers to designing the CNN architecture, including defining layers such as convolutional, pooling, and fully connected layers. This step sets up the model structure required for learning features from the images. Once the architecture is established, *"Model Training"* follows, where the model is trained on the processed dataset. This step optimizes the CNN’s weights using backpropagation and gradient descent.

A decision point, *"Define CNN", ensures the CNN structure is correctly specified before proceeding. Once confirmed, the **"Train model using model.fit()"* step executes the training process by feeding the dataset into the model, adjusting parameters to minimize loss and improve accuracy.

After training, the model is deployed using *"Deploy with Streamlit", allowing users to interact with the trained CNN through a web-based interface. This step is crucial for real-world application and usability. Before prediction, **"Image Preprocessing"* ensures that test images are standardized, resized, and formatted correctly for accurate classification.

The processed image is then passed to the *"Prediction"* stage, where the trained CNN model classifies the input as glaucomatous or non-glaucomatous. The results are then sent to the *"Display Result"* step, where Streamlit presents the outcome to the user in a user-friendly manner. Finally, the *"End"* node signifies the conclusion of the process.

CHAPTER 4

ALGORITHM

4.1 Tensorflow

TensorFlow is a widely used deep learning framework that plays a crucial role in glaucoma detection using convolutional neural networks (CNNs) and digital image classification. In this project, TensorFlow is chosen because of its efficiency in handling large-scale medical image data, its ability to accelerate deep learning computations using GPUs, and its extensive ecosystem that supports various stages of model development and deployment. CNNs are particularly effective for medical image analysis because they can automatically extract important features such as the optic cup-to-disc ratio, nerve fiber layer thickness, and other retinal abnormalities that indicate glaucoma. TensorFlow provides pre-built layers and functions that make it easier to design and train CNN architectures, allowing researchers to experiment with different models such as ResNet, VGG, and EfficientNet, which have been proven effective in medical image classification tasks.

One of the key reasons TensorFlow is used in this project is its capability to leverage transfer learning, where pre-trained models on large image datasets can be fine-tuned for glaucoma detection. This reduces the need for an extensive dataset and computational power while maintaining high accuracy. TensorFlow’s Keras API simplifies the process of building, compiling, and training deep learning models, making it easier to implement and test various architectures. Additionally, TensorFlow provides support for TensorBoard, which enables visualization of training metrics, helping in model performance analysis and optimization.

For example, in a glaucoma detection system, TensorFlow can be used to train a CNN model on fundus images to classify whether an eye is normal or affected by glaucoma. The model learns by analyzing thousands of labeled images and adjusting its parameters to improve accuracy over time. TensorFlow also supports augmentation techniques such as rotation, zooming, and flipping, which help improve the model’s generalization by making it robust to variations in input images. Furthermore, TensorFlow’s deployment options, including TensorFlow Lite and TensorFlow.js, enable real-time predictions on mobile devices and web applications, making it feasible for clinical use.

4.2 Streamlit

Streamlit is an open-source Python framework that is used in this glaucoma detection project to create an interactive and user-friendly web application for displaying and analyzing CNN-based digital image classification results. It is chosen because of its simplicity, flexibility, and ability to quickly deploy machine learning models with minimal coding effort. Since glaucoma detection involves analyzing medical images, Streamlit provides an intuitive interface where users, including doctors and researchers, can easily upload fundus or OCT images, visualize model predictions, and interact with different parameters in real time. Unlike traditional web frameworks that require extensive knowledge of front-end development, Streamlit allows seamless integration with TensorFlow models by enabling real-time inference and visualization with just a few lines of code.

One of the key advantages of using Streamlit in this project is its ability to handle dynamic updates without requiring manual page reloads. This is crucial for medical applications where real-time feedback is necessary for decision-making. For instance, once a CNN model is trained for glaucoma detection, it can be deployed within a Streamlit app, allowing users to upload new eye images and receive instant classification results, such as "Glaucoma Detected" or "Healthy Eye," along with a confidence score. Additionally, Streamlit supports various visualization libraries like Matplotlib, Seaborn, and Plotly, making it possible to display heatmaps, probability distributions, and activation maps to provide deeper insights into the model's predictions.

Another important feature of Streamlit is its built-in support for widgets such as sliders, buttons, and text inputs, which help users interact with different model parameters and fine-tune the classification thresholds. This makes it easier for researchers to experiment with different hyperparameters and analyze the model's performance without modifying the backend code. Moreover, Streamlit allows easy deployment on cloud platforms like Heroku and Streamlit Community Cloud, enabling remote access to the application without requiring users to install complex software. Overall, Streamlit enhances the usability of this glaucoma detection project by providing an interactive and visually appealing interface, making it easier for healthcare professionals and researchers to utilize deep learning models for early glaucoma diagnosis.

4.3 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are used in this glaucoma detection project because they are highly effective for image classification tasks, particularly in medical image analysis. Glaucoma is a serious eye disease that causes damage to the optic nerve, and its early detection is crucial for preventing vision loss. Traditional diagnostic methods rely on expert evaluation of retinal images, but CNNs automate this process by learning patterns and features directly from medical images. Unlike traditional machine learning models, which require manual feature extraction, CNNs can automatically detect important visual features such as the optic disc, optic cup, and retinal nerve fiber layer thickness, which are critical indicators of glaucoma.

The primary reason for using CNNs in this project is their ability to extract hierarchical features from images. The initial layers detect simple patterns like edges and textures, while deeper layers capture complex structures such as the shape and size of the optic disc. This enables CNNs to differentiate between healthy and glaucomatous eyes with high accuracy. For example, in a glaucoma detection system, a CNN model can be trained on thousands of labeled retinal fundus images to classify whether an image belongs to a normal eye or one affected by glaucoma. The model learns from the data by adjusting the weights of its neurons through backpropagation and optimization techniques, ensuring that it generalizes well to new, unseen images.

Another advantage of CNNs is their robustness to variations in lighting, orientation, and noise in medical images. By using data augmentation techniques such as rotation, zooming, and flipping, CNNs can be trained to handle diverse datasets, making them more reliable for real-world applications. Additionally, pre-trained CNN architectures like ResNet, VGG, and Inception can be used through transfer learning, reducing the need for large datasets and extensive computational resources while improving accuracy. CNNs also integrate seamlessly with frameworks like TensorFlow, allowing for efficient training and deployment in applications like Streamlit-based diagnostic tools. Overall, CNNs play a vital role in this project by providing an automated, accurate, and scalable approach to glaucoma detection, aiding in early diagnosis and improving patient outcomes.

4.4 Algorithm for training the CNN model

Step 1: Import Required Libraries

- Import TensorFlow and necessary modules for deep learning:

- Sequential for building the model.
- Conv2D, MaxPooling2D, Dense, Flatten, etc., for defining CNN layers.
- ImageDataGenerator for data augmentation.

Step 2: Initialize the Model

- Create a Sequential model.
- Add the first Convolutional Layer:
- 64 filters of size (9,9), ReLU activation, batch normalization, and max pooling.
- Add the second Convolutional Layer:
- 128 filters of size (5,5), ReLU activation, batch normalization, and max pooling.
- Add additional Convolutional Layers with increasing filters (256, 512, 512):
- Zero padding is used before convolution.
- ReLU activation and batch normalization are applied.
- Max pooling is used after some layers.

Step 3: Fully Connected Layers

- Flatten the feature maps into a 1D vector.
- Add Dense Layers:
    - A 64-unit dense layer with ReLU activation and dropout.
    - A 128-unit dense layer with ReLU activation and dropout.
    - A final output layer with 1 neuron and a sigmoid activation function (for binary classification).

Step 4: Compile the Model

- Use:
    - Binary Cross-Entropy Loss for classification.
    - Adam Optimizer for efficient learning.
    - Accuracy as the performance metric.

Step 5: Prepare Data for Training

- Use ImageDataGenerator for data augmentation:
    - Apply rescaling, rotation, shear, zoom, and horizontal flipping.
- Load the dataset from the given directory (train and val folders).
- Convert images into batches of size 64.

Step 6: Define Callbacks

- Set up callbacks:
- ReduceLROnPlateau: Reduces learning rate if validation accuracy plateaus.

- ModelCheckpoint: Saves the best model.

Step 7: Train the Model

- Train the model using .fit(), with:
    - Training data (training_set).
    - Validation data (test_set).
    - 200 epochs.
    - Defined callbacks.

Step 8: Save the Model

- Save the trained model as my_model2.h5.

4.5 Algorithm for the app

Step 1: Import Required Libraries

- Import streamlit for the web application.
- Import tensorflow for loading the trained model.
- Import PIL (Pillow) for image processing.
- Import numpy for array manipulations.

Step 2: Define the Image Preprocessing and Prediction Function

- Define import_and_predict(image_data, model):
    - Resize the image to 100×100 using ImageOps.fit().
    - Convert the image to RGB format.
    - Convert the image to a NumPy array.
    - Normalize the pixel values to the [0,1] range.
    - Expand dimensions to match the model's expected input.
    - Use model.predict() to obtain a prediction.
    - Return the prediction.

Step 3: Define the Retinal Image Validation Function

- Define is_retinal_image(image):
    - Extract image width and height.
    - Calculate aspect ratio (width/height).
    - Check if:
        1. Width and height are between 200 and 2000 pixels.
        2. Aspect ratio is approximately 1 (between 0.9 and 1.1).
    - Return True if the image passes these checks; otherwise, return False.

Step 4: Load the Trained Model

- Use tf.keras.models.load_model('my_model2.h5') to load the saved CNN model.

Step 5: Build the Streamlit Interface

- Display the application title and description using st.write().
- Show an example reference image for correct cropping.
- Provide cropping instructions for users.

Step 6: Handle Image Upload

- Use st.file_uploader() to allow users to upload a .jpg file.
- If no file is uploaded, display a message "You haven't uploaded an image file."

Step 7: Process the Uploaded Image

- If a file is uploaded:
    - Open the image using PIL.Image.open().
    - Validate if it is a retinal image using is_retinal_image().
    - If invalid, show an error message.

Step 8: Perform Prediction

- If valid, call import_and_predict(image, model).
- Extract the prediction score from the model output.
- Classify glaucoma severity based on prediction score:
    - > 0.5: Healthy
    - 0.4 - 0.5: Stage 1 Glaucoma
    - 0.3 - 0.4: Stage 2 Glaucoma
    - 0.2 - 0.3: Stage 3 Glaucoma
    - 0.1 - 0.2: Stage 4 Glaucoma
    - ≤ 0.1: Stage 5 Glaucoma
- Display the appropriate prediction result using st.write().
- If healthy, trigger a balloon animation using st.balloons().

Step 9: Handle Errors

- Use try-except blocks to catch image processing errors and model prediction errors.
- Display error messages using st.error().

CHAPTER 5

SOURCE CODE

5.1 Source code for the CNN model

```
import tensorflow as tf
from tensorflow import Sequential
from tensorflow import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
from tensorflow import ZeroPadding2D
from tensorflow import BatchNormalization
from tensorflow import ImageDataGenerator

model = Sequential()

model.add(Conv2D(64, (9, 9), input_shape=(100,100,3),padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (5, 5), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(256, (3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(512, (3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(512, (3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(128))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(1))
model.add(BatchNormalization())
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   rotation_range=30,
                                   shear_range=0.2,
                                   zoom_range=[0.8, 1.2],
                                    horizontal_flip=True,
#                                     vertical_flip = True,
                                    fill_mode='nearest')

test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
training_set = train_datagen.flow_from_directory('/content/drive/My Drive/data/train',
                                                 target_size = (100, 100),
                                                 batch_size = 64,
                                                 class_mode = 'binary')

test_set = test_datagen.flow_from_directory('/content/drive/My Drive/data/val',
                                            target_size = (100, 100),
                                            batch_size = 64,
                                            class_mode = 'binary')
my_callbacks = [
  # tf.keras.callbacks.EarlyStopping(patience=4, verbose=1),
  tf.keras.callbacks.ReduceLROnPlateau(factor=0.1, patience=3, min_lr=0.00001,
verbose=1),
  tf.keras.callbacks.ModelCheckpoint('my_model2.h5',
  verbose=1, save_best_only=True, save_weights_only=False)
  ]

model.fit(training_set, epochs=200, validation_data = test_set,
callbacks=my_callbacks)

model.save('my_model2.h5')
```

5.2 Source code for the App

```
import streamlit as st
import tensorflow as tf
from PIL import Image, ImageOps
import numpy as np

# Function to preprocess and predict the uploaded image
def import_and_predict(image_data, model):
  try:
    image = ImageOps.fit(image_data, (100, 100), Image.Resampling.LANCZOS)
    image = image.convert('RGB')
```

```
        image = np.asarray(image)
        st.image(image, channels='RGB')  # Show the uploaded image
        image = image.astype(np.float32) / 255.0
        img_reshape = np.expand_dims(image, axis=0)
        prediction = model.predict(img_reshape)
        return prediction
    except Exception as e:
        st.error(f"Error in processing the image: {e}")
        return None

# Function to check if the image is a retinal image based on its properties
def is_retinal_image(image):
    try:
        width, height = image.size
        aspect_ratio = width / height

        # A typical retinal image might have an aspect ratio close to 1 (square) and
dimensions above 200x200
        if (200 < width < 2000) and (200 < height < 2000) and (0.9 < aspect_ratio < 1.1):
            return True
        else:
            return False
    except Exception as e:
        st.error(f"Error in checking image properties: {e}")
        return False

# Load the trained model
model = tf.keras.models.load_model('my_model2.h5')

# Display header and description
st.write("""
         # ***Glaucoma Detector***
         This is a simple image classification web app to predict glaucoma through the
fundus image of the eye.
```

```
        """)
st.write("""Please upload a cropped image of your eye. The image should look like
the example below.""")

# Show reference image with cropping instructions
st.image("full to cropped.jpg", caption="Example of a cropped eye image.",
use_container_width=True)  # Provide a reference image for cropping instructions
st.write("""
        ### Instructions:
        - Crop the image so that it focuses on the eye, with minimal background.
        - The crop should ideally be centered on the eye's pupil and exclude excessive
space around the eye.
        - The reference image above shows how the crop should look.
        """)

# Upload file section
file = st.file_uploader("Please upload a .jpg image file", type=["jpg"])

if file is None:
    st.text("You haven't uploaded an image file.")
else:
    try:
        imageI = Image.open(file)  # Open the uploaded image

        # Check if the image is a retinal image
        if not is_retinal_image(imageI):
            st.error("The uploaded image does not appear to be a retinal image. Please
upload a valid retinal image.")
        else:
            prediction = import_and_predict(imageI, model)

            if prediction is not None:
                pred = prediction[0][0]
                if pred > 0.5:
```

```
                st.write("## **Prediction:** Your eye appears healthy. Great!")
                st.balloons()
            elif pred <= 0.5 and pred > 0.4:
                st.write*("## **Prediction:** Your eye appears to be at risk of stage 1
Glaucoma. Please consult an ophthalmologist.")
            elif pred <= 0.4 and pred > 0.3:
                st.write("## **Prediction:** Your eye appears to be at risk of stage 2
Glaucoma. Please consult an ophthalmologist.")
            elif pred <= 0.3 and pred > 0.2:
                st.write("## **Prediction:** Your eye appears to be at risk of stage 3
Glaucoma. Please consult an ophthalmologist.")
            elif pred <= 0.2 and pred > 0.1:
                st.write("## **Prediction:** Your eye appears to be at risk of stage 4
Glaucoma. Please consult an ophthalmologist.")
            else:
                st.write("## **Prediction:** You appear to be affected by stage 5
Glaucoma. Please consult an ophthalmologist.")
    except Exception as e:
        st.error(f"Error in loading or processing the image: {e}")
```

CHAPTER 6

FUTURE ENHANCEMENTS AND CONCLUSION

6.1 Future enhancements

While the current system provides an effective AI-based approach for glaucoma detection, several enhancements can further improve its accuracy, efficiency, and real-world applicability:

1. Larger and More Diverse Dataset – Expanding the dataset with high-quality retinal images from diverse populations will help improve the model’s generalization and reduce biases, ensuring better performance across different demographics.
2. Multi-Modal Diagnosis – Integrating additional diagnostic data such as Optical Coherence Tomography (OCT), intraocular pressure (IOP) readings, and visual field tests can enhance the system’s reliability, making it a more comprehensive glaucoma screening tool.
3. Improved Model Performance – Experimenting with advanced deep learning architectures such as Vision Transformers (ViTs), hybrid CNN-RNN models, and attention mechanisms can enhance classification accuracy and feature extraction capabilities.
4. Explainability and Interpretability – Incorporating Grad-CAM (Gradient-weighted Class Activation Mapping) or other explainability tools can help provide visual justifications for the model’s predictions, increasing trust among medical professionals.
5. Real-Time Mobile App Integration – Developing a mobile application for real-time glaucoma screening using smartphone fundus cameras will improve accessibility, particularly in remote areas with limited access to ophthalmologists.
6. Cloud-Based Deployment – Deploying the system on a cloud platform (e.g., AWS, Google Cloud, or Azure) can enable remote access, allowing healthcare professionals to upload retinal images and receive instant AI-driven analysis from anywhere.
7. Continuous Model Training – Implementing an auto-updating model that continuously learns from new patient data will help refine its accuracy over time

and adapt to evolving diagnostic standards.

8. Integration with Electronic Health Records (EHR) – Linking the AI system with hospital databases and EHR systems can enable seamless patient monitoring and facilitate long-term glaucoma management.

6.2 Conclusion

Glaucoma is a leading cause of irreversible blindness, making early detection essential for preventing vision loss. Traditional diagnostic methods, while effective, are often resource-intensive, requiring specialized equipment and trained professionals. This project addresses these challenges by utilizing deep learning and Convolutional Neural Networks (CNNs) for automated glaucoma detection from retinal fundus images.

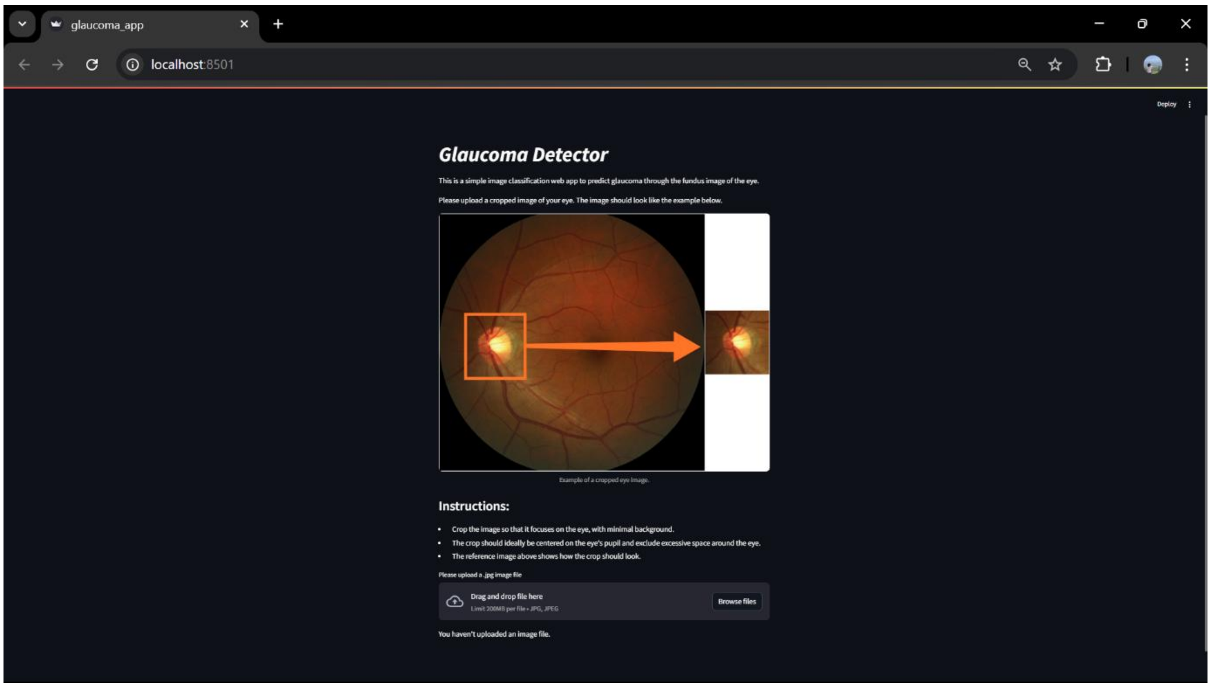
By leveraging TensorFlow, the system achieves accurate image classification, identifying glaucomatous and non-glaucomatous eyes based on retinal image features. The integration of Streamlit allows for a user-friendly web-based interface, enabling easy accessibility for both healthcare professionals and patients. This AI-powered solution offers a fast, cost-effective, and scalable approach to glaucoma screening, particularly benefiting remote and underserved regions with limited access to ophthalmic care.

The significance of this project lies in its potential to enhance early diagnosis, reduce human error, and assist ophthalmologists in prioritizing high-risk cases. Future improvements could include further refinement of the model, expansion of the dataset, and integration with other diagnostic tools to improve accuracy and reliability.

In conclusion, this project demonstrates the power of AI in revolutionizing medical diagnostics, contributing to more efficient, accessible, and proactive eye care. With continued advancements in deep learning, such technologies can play a crucial role in preventing blindness and improving global eye health.

CHAPTER 7

PROJECT GALLERY



Glaucoma Detector

This is a simple image classification web app to predict glaucoma through the fundus image of the eye.

Please upload a cropped image of your eye. The image should look like the example below.

Example of a cropped eye image.

Instructions:

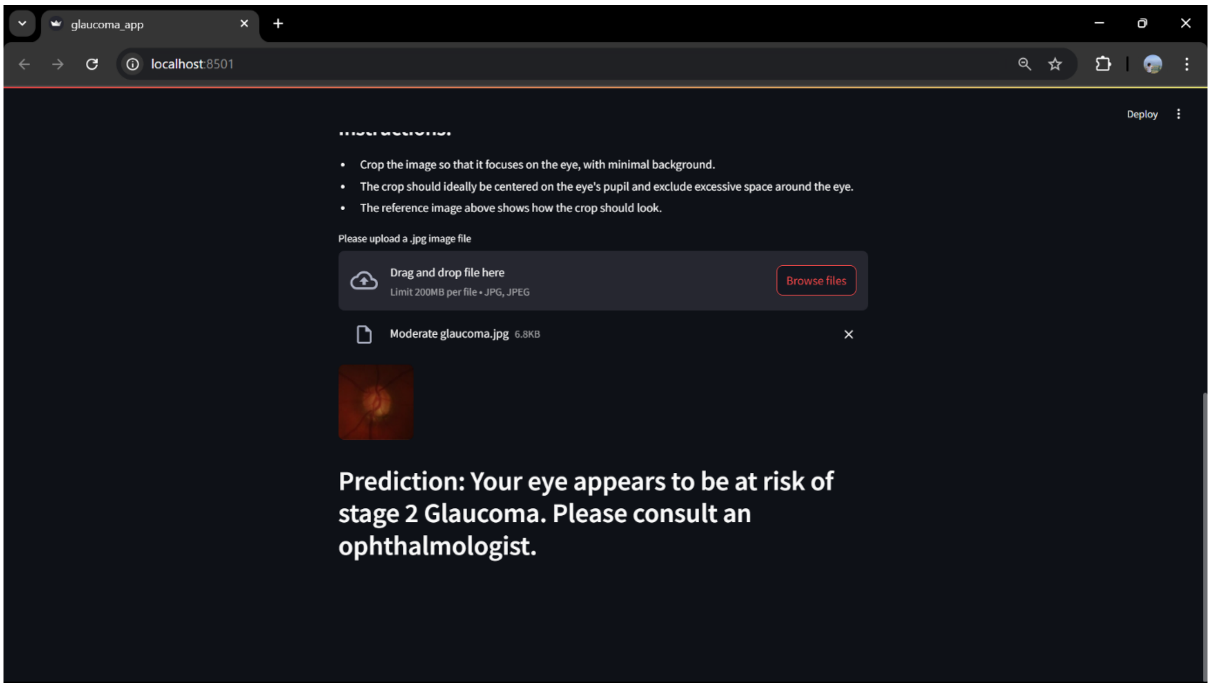
- Crop the image so that it focuses on the eye, with minimal background.
- The crop should ideally be centered on the eye's pupil and exclude excessive space around the eye.
- The reference image above shows how the crop should look.

Please upload a .jpg image file

Drag and drop file here
Limit 200MB per file • JPG, JPEG
Browse files

You haven't uploaded an image file.

Figure 1: Initial stage of the app



- Crop the image so that it focuses on the eye, with minimal background.
- The crop should ideally be centered on the eye's pupil and exclude excessive space around the eye.
- The reference image above shows how the crop should look.

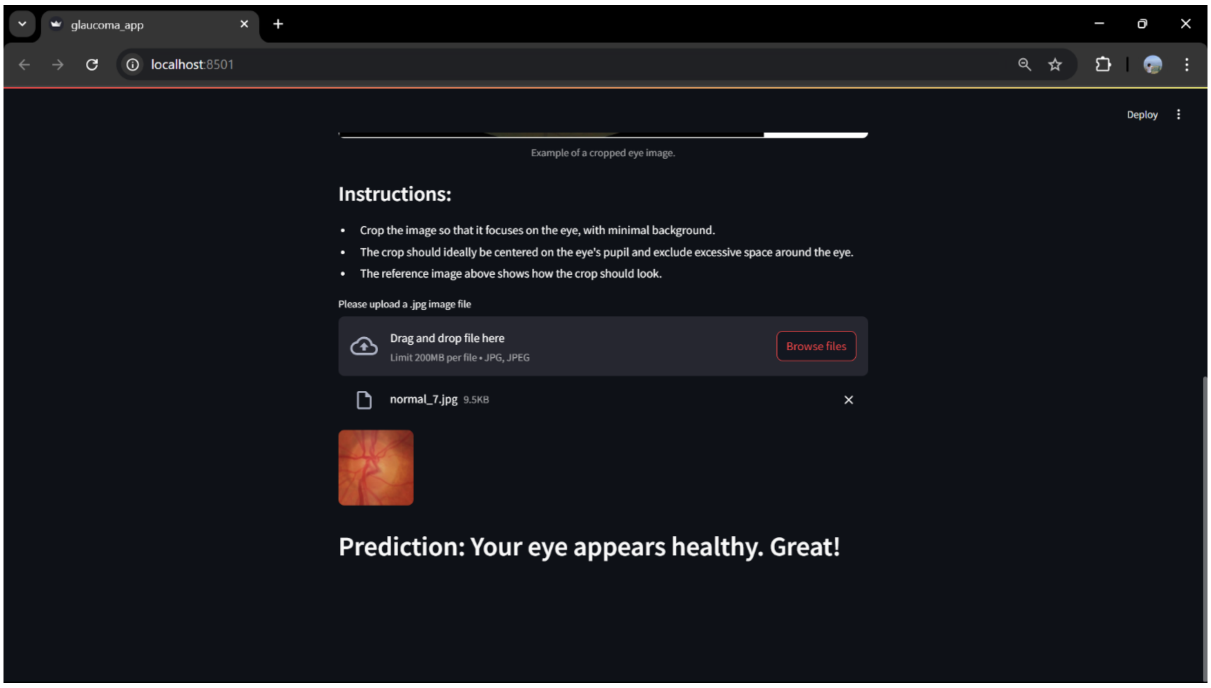
Please upload a .jpg image file

Drag and drop file here
Limit 200MB per file • JPG, JPEG
Browse files

Moderate glaucoma.jpg 6.8KB

Prediction: Your eye appears to be at risk of stage 2 Glaucoma. Please consult an ophthalmologist.

Figure 2: Glaucomatous eye



Example of a cropped eye image.

Instructions:

- Crop the image so that it focuses on the eye, with minimal background.
- The crop should ideally be centered on the eye's pupil and exclude excessive space around the eye.
- The reference image above shows how the crop should look.

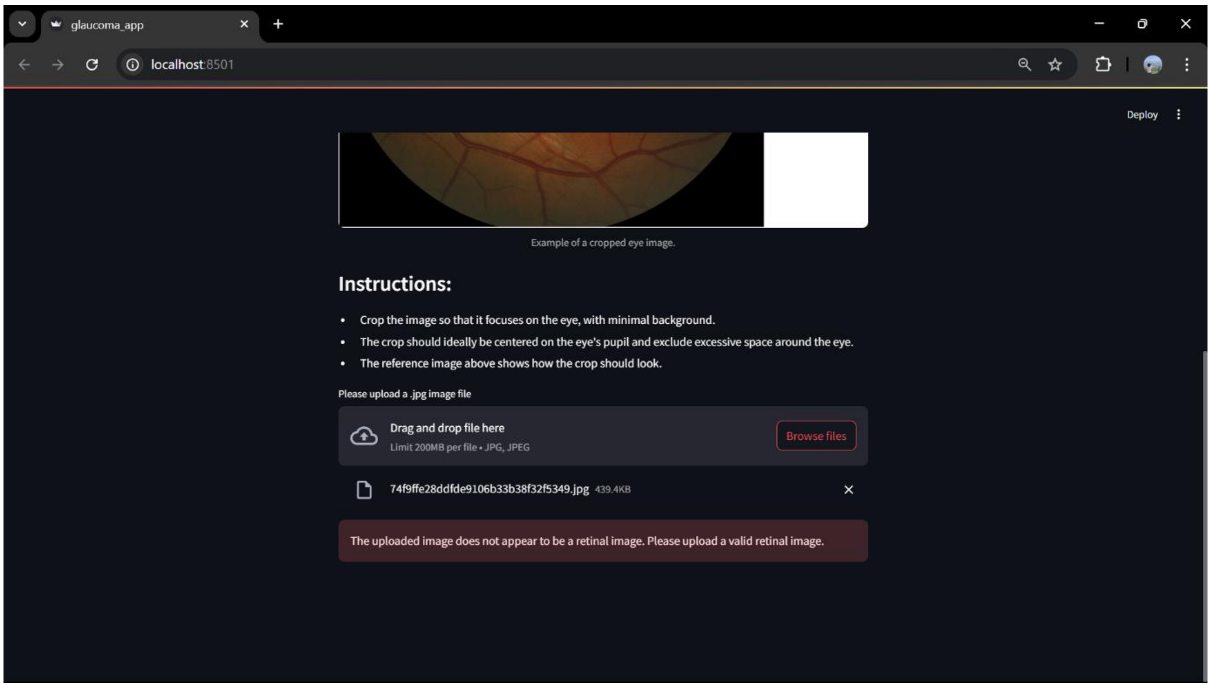
Please upload a .jpg image file

Drag and drop file here
Limit 200MB per file • JPG, JPEG
Browse files

normal_7.jpg 9.5KB

Prediction: Your eye appears healthy. Great!

Figure 3: Healthy eye



Example of a cropped eye image.

Instructions:

- Crop the image so that it focuses on the eye, with minimal background.
- The crop should ideally be centered on the eye's pupil and exclude excessive space around the eye.
- The reference image above shows how the crop should look.

Please upload a .jpg image file

Drag and drop file here
Limit 200MB per file • JPG, JPEG
Browse files

74f9ffe28ddfde9106b33b38f32f5349.jpg 439.4KB

The uploaded image does not appear to be a retinal image. Please upload a valid retinal image.

Figure 4: When uploaded image is not a retinal image

CHAPTER 8

REFERENCES

- García, G., del Amor, R., Colomer, A., & Naranjo, V. (2020). *Glaucoma detection from raw circumapillary OCT images using fully convolutional neural networks*. arXiv preprint arXiv:2006.00027. https://arxiv.org/abs/2006.00027
- Maheshwari, S., Kanhangad, V., & Pachori, R. B. (2020). *CNN-based approach for glaucoma diagnosis using transfer learning and LBP-based data augmentation*. arXiv preprint arXiv:2002.08013. https://arxiv.org/abs/2002.08013
- Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). *Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs*. JAMA, 316(22), 2402–2410. https://doi.org/10.1001/jama.2016.17216
- Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). *Dermatologist-level classification of skin cancer with deep neural networks*. Nature, 542(7639), 115–118. https://doi.org/10.1038/nature21056
- Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C. S., Liang, H., Baxter, S. L., ... & Zhang, K. (2018). *Identifying medical diagnoses and treatable diseases by image-based deep learning*. Cell, 172(5), 1122–1131. https://doi.org/10.1016/j.cell.2018.02.010
- Li, Z., He, Y., Keel, S., Meng, W., Chang, R. T., & He, M. (2018). *Efficacy of a deep learning system for detecting glaucomatous optic neuropathy based on color fundus photographs*. Ophthalmology, 125(8), 1199–1206. https://doi.org/10.1016/j.ophtha.2018.02.022
- Christopher, M., Belghith, A., Bowd, C., Goldbaum, M. H., Weinreb, R. N., Fazio, M. A., ... & Girkin, C. A. (2020). *Performance of deep learning architectures and transfer learning for detecting glaucomatous optic neuropathy in fundus photographs*. Scientific Reports, 10(1), 1–13. https://doi.org/10.1038/s41598-020-65010-w
- Fu, H., Cheng, J., Xu, Y., Zhang, C., Wong, D. W. K., Liu, J., & Cao, X. (2018). *Disc-aware ensemble network for glaucoma screening from fundus images*. IEEE Transactions on Medical Imaging, 37(11), 2493–2501. https://doi.org/10.1109/TMI.2018.2837027
- Chen, X., Xu, Y., Wong, D. W. K., Wong, T. Y., & Liu, J. (2015). *Glaucoma*

detection based on deep convolutional neural network. In *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 715–718). IEEE. https://doi.org/10.1109/EMBC.2015.7318431

- An, G., Yu, J., & Lu, W. (2019). *Glaucoma detection based on deep learning network in fundus image*. In *2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)* (pp. 79–83). IEEE. https://doi.org/10.1109/ICSIP.2019.8836622
- Shibata, Y., & Kawasaki, R. (2020). *Application of artificial intelligence in ophthalmology*. Asia-Pacific Journal of Ophthalmology, 9(4), 299–307. https://doi.org/10.22608/APO.2020185
- Ting, D. S., Cheung, C. Y., Lim, G., Tan, G. S., Quang, N. D., Gan, A., ... & Wong, T. Y. (2017). *Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes*. JAMA, 318(22), 2211–2223. https://doi.org/10.1001/jama.2017.18152
- Mookiah, M. R. K., Acharya, U. R., Lim, C. M., Petznick, A., & Suri, J. S. (2013). *Data mining technique for automated diagnosis of glaucoma using higher-order spectra and wavelet energy features*. Knowledge-Based Systems, 33, 73–82. https://doi.org/10.1016/j.knosys.2012.12.014
- Chen, X., Xu, Y., Wong, D. W. K., Wong, T. Y., & Liu, J. (2015). *Glaucoma detection based on deep convolutional neural network*. In *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 715–718). IEEE. https://doi.org/10.1109/EMBC.2015.7318431
- An, G., Yu, J., & Lu, W. (2019). *Glaucoma detection based on deep learning network in fundus image*. In *2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)* (pp. 79–83). IEEE. https://doi.org/10.1109/ICSIP.2019.8836622